LEARNING TO FORGET TOGETHER: PROMPT-CONDITIONED ADAPTER GENERATION FOR MULTI-DOMAIN LLM UNLEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) can memorize and disclose sensitive information, making machine unlearning an important tool for mitigating privacy and safety risks. However, removing knowledge across multiple domains can cause interference between unlearning updates and degrade general model capabilities. The need to retrain for different combinations of target domains further complicates deployment. We introduce PUMA (Prompt-conditioned Unlearning via Multi-domain Adapter Generation), a framework that generates low-rank adaptation (LoRA) weights from prompts representing the knowledge to be forgotten. A shared parameter generator learns to translate instance-level prompts into unlearning adapters, enabling flexible requests expressed through examples. Once trained, PUMA supports instance-based, multi-domain unlearning without further optimization for each request, enabling zero-shot deployment. It combines forgetting objectives across domains within a single generated adapter, eliminating the need to train a separate adapter for every domain combination. Experiments show that PUMA achieves a better balance between forgetting efficacy and retained utility than existing domain-level unlearning methods while accommodating multiple target domains. These results highlight the potential of prompt-conditioned parameter generation for flexible and efficient LLM unlearning.

1 INTRODUCTION

Large language models (LLMs) acquire broad capabilities from large-scale pretraining, but can also memorize and disclose sensitive information or retain knowledge that enables harmful use. Addressing these risks requires a way to suppress selected knowledge after training while preserving the model's general capabilities. Machine unlearning (MU) provides the foundations for removing the influence of designated data without repeating the entire training process (Ginart et al., 2019; Guo et al., 2019; Bourtoule et al., 2021; Sekhari et al., 2021). Its application to LLMs has made substantial progress in reducing access to unwanted information (Jang et al., 2023; Yao & Xu, 2024), with objectives such as NPO and SimNPO improving the balance between forgetting and retained utility (Zhang et al., 2024; Fan et al., 2026). Benchmarks including TOFU, RWKU, and WMDP further establish the importance of evaluating these methods on memorized, real-world, and hazardous knowledge (Maini et al., 2024; Cao et al., 2024; Li et al., 2024b).

In practical applications, however, an unlearning system must accommodate requests involving multiple domains within a common deployment. A solution that works for one isolated domain may become less effective when different forgetting objectives share model parameters. Existing approaches explore combining lightweight unlearning layers, aggregating parameter updates, and processing requests through continual unlearning (Chen & Yang, 2023; Ilharco et al., 2022; Xu et al., 2026a). LoRA makes individual updates inexpensive to parameterize (Hu et al., 2021), but combining domain-specific updates can weaken target suppression or damage unrelated capabilities, while repeated updates can accumulate interference. These difficulties make retained utility as important as forgetting efficacy in multi-domain evaluation (Shi et al., 2025). Training and storing a separate adapter for every domain provides a strong reference, yet leaves the relationship between a textual request and its appropriate update to a separate selection mechanism. The remaining challenge

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

is to consolidate forgetting behavior across domains while preserving the selectivity of individual requests.

To address this challenge, we introduce **PUMA**, **P**rompt-conditioned **U**nlearning via **M**ulti-domain **A**dapter Generation, a paradigm that learns to generate unlearning parameters from text. Building on hypernetworks and prompt-conditioned parameter generation (Ha et al., 2017; Phang et al., 2023; Liang et al., 2026), PUMA trains a single shared generator using LoRA checkpoints collected from multiple unlearning domains. During training, representative texts are paired with checkpoints from the corresponding domain, and the generator learns to reconstruct their adapter weights. A frozen text encoder supplies the conditioning features, while structural embeddings and module-specific residual heads help the decoder account for the organization of LoRA parameters. At deployment, the generator maps the examples in a request directly to an adapter for the frozen LLM, without further gradient optimization. This establishes an *implicit route from domain-relevant text to LoRA weights*: the textual condition determines the generated update without requiring an explicit domain classifier or a discrete adapter-selection rule. Joint training allows PUMA to learn shared structure from unlearning adapters across domains, while textual conditions guide which forgetting behavior to generate for each request. This combines information across domains with an instance-based interface for specifying the target, enabling generated adapters to benefit from more than a single domain's training trajectory. Once trained, the generator produces the corresponding LoRA weights without further gradient optimization.

Experiments across multiple forgetting domains and LLM backbones support this approach. PUMA achieves both stronger average forgetting and higher retained utility than Single, the gold reference that trains a separate LoRA for each domain. Component ablations demonstrate the benefits of structural embeddings and module-specific refinement, while increasing the number of training domains preserves performance on the evaluated targets. Visualizations reveal domain-organized generated updates and an association between text similarity and update similarity, supporting the role of textual conditioning in implicit routing. Cross-domain evaluations further show reduced collateral degradation in several non-requested domains. These findings support text-conditioned parameter generation as an effective paradigm for consolidating multi-domain unlearning while preserving request-dependent behavior.

Our contributions are threefold:

- We identify the unresolved tension between forgetting efficacy and retained utility when accommodating multiple domains, and examine the limitations of composing or jointly optimizing domain-level unlearning updates.
- We introduce PUMA, a shared text-conditioned LoRA generator that learns from multi-domain unlearning checkpoints and implicitly routes instance-level textual conditions to request-specific adapters without per-request gradient optimization.
- We demonstrate improved average forgetting and retained utility over independently trained domain-specific LoRA adapters across four backbones, with component ablations and cross-domain diagnostics supporting the effectiveness of the proposed paradigm.

2 Related Work

Machine unlearning in LLMs. Machine unlearning addresses the privacy and safety risks of knowledge retained by LLMs. Early methods suppress unwanted information through gradient-based updates (Jang et al., 2023; Yao & Xu, 2024), while NPO and SimNPO improve the balance between forgetting and utility (Zhang et al., 2024; Fan et al., 2026); in-context unlearning instead guides behavior through supplied examples (Pawelczyk et al., 2023). TOFU and RWKU evaluate selective forgetting (Maini et al., 2024; Cao et al., 2024), and broader evaluations examine utility preservation and the robustness of suppression (Shi et al., 2025; Dorna et al., 2026; Tamirisa et al., 2025). Extending these advances to multiple domains requires managing interactions among different forgetting requests. EUL combines lightweight unlearning layers through learned fusion (Chen & Yang, 2023), and FIT addresses utility degradation and knowledge recovery under continual deletion requests (Xu et al., 2026a). These approaches make progress in accommodating multiple requests, but preserving selective forgetting as updates are combined or accumulated remains a central challenge. Selectivity also depends on the granularity of the target: domain-based

unlearning suppresses a body of knowledge, such as a fictional universe or hazardous subject area (Eldan & Russinovich, 2023; Li et al., 2024b), whereas instance-based unlearning targets specified statements or passages. Targeted unlearning and BiForget study this finer-grained distinction (Liu et al., 2024; Xu et al., 2026b). PUMA connects multi-domain supervision with an instance-based conditioning interface: a shared generator learns from domain-specific LoRA checkpoints and maps representative texts to the corresponding adapter. This provides implicit routing from a request to its unlearning weights, with cross-domain integration occurring in generator training. The examples condition the generated behavior; they do not by themselves establish exact record-level deletion.

Neural parameter generation. Hypernetworks learn to generate the parameters of a target network, turning weight construction into a learned mapping (Ha et al., 2017). Early work on parameter prediction and neural-network generation established the feasibility of learning in weight space (Denil et al., 2013; Deutsch, 2018). Subsequent advances model training checkpoints with generative networks (Peebles et al., 2022), while Neural Network Diffusion and recurrent diffusion extend this approach to larger parameter spaces (Wang et al., 2024; 2025). These developments support applications in language-model adaptation: shared hypernetworks generate task-conditioned adapters for multi-task learning (Mahabadi et al., 2021), HyperTuning produces adaptation parameters from examples (Phang et al., 2023), and DnD maps prompts to LoRA weights through a text encoder and hyper-convolutional decoder (Liang et al., 2026; Hu et al., 2021). Related applications of learned parameter transformations include model editing, where MEND transforms gradients into targeted updates (Mitchell et al., 2021); model soups and task arithmetic provide complementary ways to reuse trained weights through averaging and composition (Wortsman et al., 2022; Ilharco et al., 2022). Parameter generation has also been applied to machine unlearning. HyperForget uses diffusion hypernetworks to generate models conditioned on forgetting requests (Rangel et al., 2024), and UnHype generates CLIP-conditioned LoRA weights for concept erasure in text-to-image models (Wójcik et al., 2026). Building on these advances, PUMA applies text-conditioned parameter generation to multi-domain LLM unlearning. It jointly learns from domain-specific LoRA checkpoints, using a shared generator to map request examples to unlearning adapters and implicitly route between domains.

3 METHOD

PUMA trains a one-for-all parameter generator on unlearning checkpoints from multiple domains. The same generator maps representative texts from each target domain to its corresponding LoRA adapter, consolidating domain-specific adapter generation into a shared model. As shown in Figure 1, the pipeline has three stages: collecting domain-specific reference adapters, training a prompt-conditioned parameter generator, and generating LoRA adapters for deployment.

3.1 PARAMETER GENERATOR DESIGN

PUMA learns a shared mapping from representative texts to unlearning adapters, allowing the same generator to serve requests across domains. Let p_{θ_0} be a pretrained causal language model and $\mathcal{D}$ the set of target domains, each associated with a forget corpus F_d and a retain corpus R_d. Given an ordered group of texts $S_d = (x_1, \ldots, x_m)$ representing a request in domain d, the generator produces LoRA factors $\widehat{\phi}_d$. These factors define an update to each adapted linear module ℓ:

$$W'_\ell = W_{0,\ell} + \Delta W_\ell(\widehat{\phi}_d), \qquad \Delta W_\ell(\phi) = \frac{\alpha_\ell}{r_\ell} B_\ell A_\ell, \tag{1}$$

where $A_\ell \in \mathbb{R}^{r_\ell \times n_\ell}$ and $B_\ell \in \mathbb{R}^{o_\ell \times r_\ell}$ are the low-rank factors, r_ℓ is the adapter rank, and α_ℓ controls its scaling. The backbone weights remain frozen, while the supplied texts determine the adapter generated for each request.

The generator first encodes the input texts into token-level features using a frozen encoder E. A hyper-convolutional decoder H_η then maps these features to an initial parameter tensor $Z^{(0)} = H_\eta(E(S))$. This tensor is organized into N chunks of shape $h \times w$, with metadata for reconstructing the LoRA factors. Although the decoder is shared, its outputs correspond to parameters with distinct locations and roles in the backbone. We account for this structure through two complementary components: structural embeddings encode each chunk's location, and module-specific residual

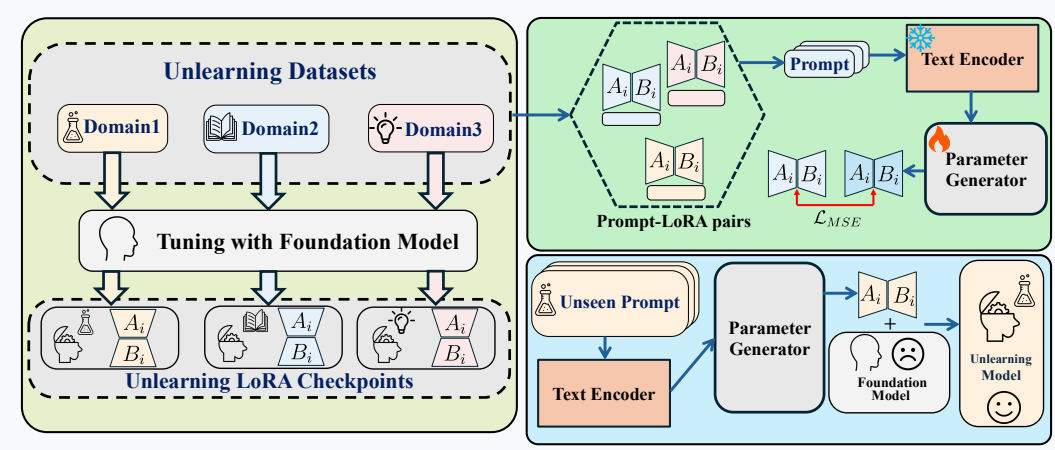


Figure 1: Overview of PUMA. Left: domain-specific unlearning produces LoRA checkpoints, which are paired with representative texts to form a multi-domain supervision bank. Top right: a frozen text encoder conditions a shared parameter generator, trained to reconstruct the paired LoRA parameters. Bottom right: a new prompt is encoded to generate an adapter for the frozen foundation model without per-request gradient updates. Unseen prompts refer to held-out conditioning texts, not necessarily unseen domains.

heads refine its values according to the projection it belongs to. Together with the decoder, these components form the generator G_ψ.

Structural Embeddings. Each generated chunk belongs to a particular layer, projection module, and LoRA factor. We make this information available to the decoder through a learned representation of the chunk's position in the adapter. Specifically, chunk j is indexed by its layer l_j, module type u_j, factor identity $f_j \in \{A, B\}$, and position p_j within that factor. Embeddings of these indices are combined in a shared d_s-dimensional space:

$$e_j = e^{\text{layer}}_{l_j} + e^{\text{module}}_{u_j} + e^{\text{factor}}_{f_j} + e^{\text{chunk}}_{p_j}. \tag{2}$$

A shared projection $P \in \mathbb{R}^{hw \times d_s}$ maps this representation to a correction with the same shape as the predicted chunk:

$$\widetilde{Z}_j = Z^{(0)}_j + \text{reshape}_{h\times w}(Pe_j). \tag{3}$$

This correction captures information about parameter location that is shared across requests, complementing the text-dependent prediction $Z^{(0)}_j$. The embedding tables follow the adapter layout of the target backbone. We initialize P to zero so that the branch initially preserves the decoder's predictions and learns structural corrections during training.

Module-Specific Residual Heads. Structural embeddings describe where a chunk belongs, but attention and feed-forward projections also differ in their functional roles. To accommodate these differences, we refine the corrected chunks with a residual head for each module type $u \in \{q, k, v, o, \text{gate}, \text{up}, \text{down}\}$. For a row vector $z \in \mathbb{R}^w$, the refinement is

$$R_u(z) = W_{u,2}\,\text{SiLU}(W_{u,1}\,\text{LN}(z) + b_{u,1}) + b_{u,2}, \tag{4}$$

where $W_{u,1} \in \mathbb{R}^{d_h \times w}$ and $W_{u,2} \in \mathbb{R}^{w \times d_h}$ define a bottleneck of width d_h. Layer normalization operates along the last dimension without learned affine parameters. Each head acts row-wise and is shared across layers and LoRA factors of the same module type. Applying the appropriate head to each structurally corrected chunk yields

$$\widehat{Z}_j = \widetilde{Z}_j + R_{u_j}(\widetilde{Z}_j), \qquad \widehat{Z} = G_\psi(E(S)). \tag{5}$$

The residual formulation allows each module type to refine the shared decoder's predictions through a specialized transformation. As with the structural branch, the output projection and bias of each head are initialized to zero, preserving the original decoder output at initialization. Both components are trained jointly with the decoder. Backbone-specific dimensions and implementation settings are detailed in Appendix B.

3.2 ONE-FOR-ALL TRAINING AND INFERENCE

We train the shared generator to map representative texts to unlearning adapters using supervision collected across domains. This process first constructs a bank of domain-specific LoRA checkpoints, then learns a common text-to-parameter mapping from their pooled representations. Once trained, the generator produces an adapter directly from the examples supplied with a request.

Constructing the reference adapter bank. To provide the generator with examples of domain-specific forgetting, we train reference adapters with an EUL-inspired objective that balances target suppression and knowledge retention (Chen & Yang, 2023). Let $C_X(\phi)$ denote the causal language-model cross-entropy on corpus X. We measure the adapter's departure from the frozen reference model through

$$K_X(\phi) = \mathbb{E}_{x\sim X,t}\left[\mathrm{KL}(p_{\theta_0}(\cdot \mid x_{<t}) \,\|\, p_{\theta_0,\phi}(\cdot \mid x_{<t}))\right] \tag{6}$$

where the expectation averages next-token divergence over non-padding positions. The forget and retain objectives for domain d are

$$\mathcal{L}_{f,d} = -K_{F_d} - \gamma_d C_{F_d}, \qquad \mathcal{L}_{r,d} = a_d K_{R_d} + \lambda_d C_{R_d}. \tag{7}$$

Minimizing $\mathcal{L}_{f,d}$ encourages the adapted model to depart from the reference predictions and reduce the likelihood of the forget texts. The retain objective instead encourages agreement with the reference and accurate prediction on retained data. The coefficients γ_d, a_d, and λ_d control the relative contributions of these terms. We alternate optimization steps between the objectives, updating only the LoRA factors while keeping the backbone frozen.

After domain-specific training, we continue optimization at a reduced learning rate and collect a sequence of checkpoints $\Phi_d = \{\phi_{d,k}\}_{k=1}^{K}$. These snapshots provide the generator with parameter targets along each domain's unlearning trajectory. They come from a shared optimization run within that domain; the checkpoint collection settings are given in Appendix A.

Multi-domain checkpoint supervision. We pool the domain-specific checkpoints into a supervision bank $\mathcal{B} = \bigcup_{d\in\mathcal{D}}\{(d,\phi) : \phi \in \Phi_d\}$ and train a single generator across this bank. Each training pair consists of a checkpoint $\phi_{d,k}$ and a group of texts sampled without replacement from the same domain's conditioning pool. Varying both the checkpoint and the text group exposes the generator to different parameter targets and textual descriptions of each domain. Domain labels establish the training pairs, while the generator itself is conditioned on the text features.

A parameter tokenizer $\mathcal{T}$ converts the reference LoRA factors into the chunk layout predicted by the generator, giving

$$\widehat{Z} = G_\psi(E(S_d)), \qquad Z_{d,k} = \mathcal{T}(\phi_{d,k}). \tag{8}$$

Each chunk is standardized, and its mean and standard deviation are stored as boundary metadata so that the original factor values can be recovered. Together with padding, these chunks form a target tensor of shape $N \times h \times w$. Normalization and metadata decoding use high precision to preserve small scale statistics.

The generator learns to reconstruct these targets through a weighted, masked mean-squared error. Let M_i indicate whether entry i is valid and $w_{c(i)}$ denote the weight assigned to its token channel. The training objective is

$$\mathcal{L}_{\mathrm{gen}}(\psi) = \mathbb{E}_{d,k,S_d}\left[\frac{\sum_i M_i w_{c(i)}(\widehat{Z}_i - Z_{d,k,i})^2}{\sum_i M_i}\right]. \tag{9}$$

The mask excludes padded entries, while the channel weights reflect variation across the reference checkpoints. We compute these weights from token standard deviations across checkpoints, normalize them within each domain, and average them across domains. This objective directly supervises parameter generation without evaluating or backpropagating through the base LLM. The decoder, structural embeddings, and residual heads are trained jointly, while the text encoder remains frozen and in evaluation mode. Shared training integrates supervision across domains, and the textual conditions guide which adapter behavior the generator learns to produce.

Generating unlearning adapters. At deployment, the same encoder and generator transform a request's representative texts S into parameter tokens. Inverting the tokenizer recovers the corresponding LoRA factors:

$$\widehat{\phi}(S) = \mathcal{T}^{-1}(G_\psi(E(S))) . \tag{10}$$

The factors are then attached to the frozen backbone using Equation 1. The text-to-parameter mapping provides implicit routing from the supplied examples to a domain-relevant update, without a separate adapter-selection step. Adapter generation requires no further gradient optimization for an individual request. This is the sense in which PUMA supports zero-shot deployment with instance-based conditioning.

4 Experiment

4.1 Implementation Details

Models. We evaluate PUMA on Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, Qwen3-8B, and Llama-3.1-8B-Instruct, covering different model sizes and families. The Qwen results are presented in the main comparison, with Llama results included in the appendix. Each backbone has its own reference LoRA bank and parameter generator. We start from the released model checkpoints without an additional supervised fine-tuning stage to teach the target knowledge before unlearning. During source-adapter training, only the LoRA parameters are updated; during generator training, supervision comes from the saved adapters rather than gradients through the backbone.

Datasets. We use MMLU (Hendrycks et al., 2020) to evaluate forgetting of knowledge already present in the starting models. Unlike a memorize-then-forget pipeline that first introduces a designated corpus through supervised fine-tuning, this setup measures how unlearning changes the model's existing subject knowledge. It also provides a common benchmark for evaluating both targeted suppression and preservation of other capabilities. We select eight MMLU subjects as forgetting domains: Prehistory, Professional Psychology, High School World History, Philosophy, Human Sexuality, Nutrition, Sociology, and Security Studies. The remaining 49 subjects form a fixed retained evaluation set. We additionally include a combined Biology and Cybersecurity domain from WMDP (Li et al., 2024b) to assess hazardous-knowledge suppression; chemistry is excluded from the reported forgetting and retention aggregates.

For adapter training and generator conditioning, we construct domain-specific text corpora separately from the multiple-choice evaluation format. The MMLU-domain corpora organize subject knowledge into topic-based explanatory passages, rather than using the official evaluation questions as training examples. The data preparation scripts render these passages from domain topic material and check their length and repetitive structure. The WMDP domain uses the corresponding forgetting corpora, while the reference-adapter recipe draws retention texts from the economics, law, and physics corpora supplied with the WMDP data pipeline. These training retention corpora are distinct from the Retain49 evaluation set. The default condition split reserves 100 texts per domain for generation, leaving 400 training conditions for a 500-text pool. This split holds out conditioning records within supervised domains; dataset construction and the limitations of textual overlap are detailed in Appendix C.

Metrics. We report forgetting-domain accuracy and retained utility, both as percentages. For an MMLU forgetting domain d, the forgetting score is the accuracy on its corresponding subject. For the combined WMDP domain, it is the unweighted mean of Biology and Cybersecurity accuracy. Lower target accuracy indicates stronger suppression and is interpreted together with retained utility, since a general loss of model capability can also reduce target accuracy. When reporting an overall forgetting score, we average the nine domain scores equally, so the combined WMDP domain contributes one term.

Let $A_j(M)$ be model M's accuracy on task j, $\mathcal{J}$ the 57 MMLU subjects, and $\mathcal{F}$ the eight MMLU forgetting subjects. Retained utility is the unweighted task average over $\mathcal{R} = \mathcal{J} \setminus \mathcal{F}$:

$$U(M) = \frac{1}{49} \sum_{j \in \mathcal{R}} A_j(M). \tag{11}$$

For methods evaluated as a single model, Retain is $U(M)$. Single and PUMA instead provide a separate adapter M_d for each requested domain. Their forgetting column d evaluates that adapter on its target, while their reported retention is

$$\text{Retain} = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} U(M_d). \tag{12}$$

Thus, this score averages retention across domain-specific requests and does not describe a single jointly unlearned model. The same 49 retained tasks are used for all methods. Four-choice chance accuracy provides a reference for interpreting target suppression, rather than a guarantee of knowledge removal.

Training Procedure. We first train an independent reference LoRA for each domain by alternating the forgetting and retention objectives in Section 3.2. After this stage, we continue optimization at a reduced learning rate and collect the first 100 continuation checkpoints per domain. These are snapshots along each domain's training trajectory, giving the generator multiple parameter targets without requiring independent adapter-training runs for every target. We then pool the checkpoints across domains. Each generator-training example pairs a checkpoint with a group of 64 texts sampled from the corresponding domain's training-condition pool. The generator learns to reconstruct the checkpoint's parameter tokens through the masked, weighted reconstruction objective, with the text encoder frozen and the decoder and structural components trained jointly. The default generator schedule uses 5,000 optimizer steps and saves checkpoints every 1,000 steps. At inference, a group from the reserved conditioning pool is passed through the trained generator to produce a LoRA adapter for evaluation, without further gradient updates. Optimization settings and recorded deployment configurations are provided in Appendices A and A.

4.2 Main Results

We compare PUMA with Base, EUL, Arithmetic, TIES, MixLoRA, ALKN, and O^3, using independently trained domain-specific LoRA adapters (Single) as the gold reference. Across all evaluated backbones, PUMA supports one-for-all unlearning through a shared generator that produces domain-relevant adapters from instance-level textual conditions. This consolidates the generation of different forgetting behaviors without additional gradient optimization for each request. The results in Table 1, together with the additional-backbone evaluation in the appendix, show that PUMA goes beyond matching Single: it achieves lower average forgetting-domain accuracy and higher retained utility on every backbone. These gains indicate that learning jointly from domain-specific checkpoints can yield more effective adapters than using each independently trained reference directly.

The comparison also highlights why low forgetting-domain accuracy alone is insufficient to establish successful unlearning. ALKN and Arithmetic achieve low target scores in several settings, but their retained utility deteriorates sharply as well. In particular, the near-chance retention observed for ALKN is consistent with broad capability collapse rather than selective knowledge suppression. Such updates compromise the model's usefulness beyond the requested domains, making their apparent forgetting gains unsuitable as evidence of effective selective unlearning. The opposite pattern is also problematic: O^3 preserves relatively high retained utility, yet leaves substantially higher target accuracy than PUMA. Preserving general capabilities while retaining much of the knowledge designated for forgetting does not satisfy the unlearning objective either.

PUMA achieves a more favorable balance between these outcomes. Relative to methods with severe retention degradation, it preserves substantially more general capability; relative to methods with weak target suppression, it achieves stronger forgetting. Its consistent improvement over Single on both aggregate forgetting and retention further shows that this balance can be maintained within a shared generation framework. The results support combining multi-domain checkpoint supervision with instance-based textual conditioning to integrate information across adapters while preserving the distinctions between requests.

4.3 Ablation Study and Discussion

Effect of generator components. We examine structural embeddings (SE) and module-specific residual heads (MRH) by adding each component to the shared decoder separately and then combining them. In Table 2, Base denotes the unaugmented generator. Both SE and MRH reduce

Table 1: Forgetting and retained utility across nine target domains for Qwen2.5-3B, Qwen2.5-7B, and Qwen3-8B (accuracy, %). Retain averages the 49 MMLU tasks outside the eight forgotten MMLU domains. *Single* use one adapter per domain thus serves as the gold baseline; their Retain score additionally averages over the nine adapters. The Biology and Cybersecurity column averages the two corresponding WMDP tasks. Measured scores use the full benchmark.

	Forget domains ↓									Retain ↑
Method	Prehistory	Professional Psychology	High School World History	Philosophy	Human Sexuality	Nutrition	Sociology	Security Studies	Biology and Cybersecurity	Average
Qwen2.5-3B										
Base	73.5 ± 0.3	71.9 ± 0.2	85.2 ± 0.4	70.7 ± 0.5	75.6 ± 0.1	71.9 ± 0.3	83.1 ± 0.3	74.7 ± 0.2	57.4 ± 0.2	65.6 ± 0.1
EUL	34.0 ± 0.2	31.7 ± 0.4	21.1 ± 0.3	39.2 ± 0.1	37.4 ± 0.2	39.9 ± 0.2	42.8 ± 0.4	42.0 ± 0.3	26.0 ± 0.2	45.4 ± 0.3
Arithmetic	31.5 ± 0.2	27.8 ± 0.1	21.9 ± 0.3	27.3 ± 0.2	36.6 ± 0.4	34.3 ± 0.3	25.4 ± 0.2	31.0 ± 0.2	25.6 ± 0.3	31.3 ± 0.3
TIES	30.6 ± 0.1	30.1 ± 0.3	24.5 ± 0.2	39.6 ± 0.4	35.9 ± 0.2	41.8 ± 0.2	31.8 ± 0.3	33.1 ± 0.3	28.3 ± 0.2	37.6 ± 0.1
MixLoRA	39.8 ± 0.2	33.0 ± 0.3	25.7 ± 0.1	49.8 ± 0.2	42.0 ± 0.2	34.3 ± 0.1	42.3 ± 0.3	30.2 ± 0.3	27.6 ± 0.2	45.1 ± 0.1
ALKN	23.1 ± 0.3	25.0 ± 0.2	28.7 ± 0.3	21.2 ± 0.1	21.4 ± 0.4	22.2 ± 0.1	28.9 ± 0.1	25.3 ± 0.2	26.5 ± 0.3	23.4 ± 0.3
O^3	45.4 ± 0.3	52.7 ± 0.2	61.4 ± 0.2	46.5 ± 0.3	52.3 ± 0.4	59.1 ± 0.2	62.2 ± 0.3	58.3 ± 0.3	42.2 ± 0.2	62.8 ± 0.3
PUMA	26.5 ± 0.3	40.9 ± 0.2	42.0 ± 0.2	30.3 ± 0.3	35.1 ± 0.1	54.3 ± 0.2	46.2 ± 0.3	42.7 ± 0.2	36.8 ± 0.1	61.4 ± 0.3
Single	29.3 ± 0.2	41.5 ± 0.3	42.6 ± 0.2	31.8 ± 0.3	38.9 ± 0.4	55.6 ± 0.2	47.3 ± 0.2	44.9 ± 0.4	37.1 ± 0.3	59.4 ± 0.1
Qwen2.5-7B										
Base	81.8 ± 0.2	76.0 ± 0.4	87.8 ± 0.2	73.6 ± 0.2	80.9 ± 0.3	78.8 ± 0.1	90.0 ± 0.1	77.6 ± 0.3	64.9 ± 0.1	72.4 ± 0.2
EUL	34.2 ± 0.1	32.4 ± 0.2	24.2 ± 0.3	41.2 ± 0.2	44.4 ± 0.2	42.5 ± 0.3	47.2 ± 0.2	41.2 ± 0.3	25.3 ± 0.2	52.2 ± 0.6
Arithmetic	35.3 ± 0.2	28.3 ± 0.3	24.4 ± 0.2	26.3 ± 0.5	39.7 ± 0.1	39.4 ± 0.2	28.3 ± 0.1	31.1 ± 0.1	29.5 ± 0.3	37.4 ± 0.2
TIES	36.3 ± 0.2	34.2 ± 0.2	27.3 ± 0.1	45.3 ± 0.1	35.2 ± 0.2	47.1 ± 0.2	36.3 ± 0.1	36.2 ± 0.3	35.1 ± 0.2	46.2 ± 0.1
MixLoRA	45.3 ± 0.2	36.4 ± 0.3	24.5 ± 0.3	53.2 ± 0.2	47.1 ± 0.2	38.1 ± 0.2	47.2 ± 0.2	34.2 ± 0.1	33.3 ± 0.2	46.8 ± 0.2
ALKN	26.3 ± 0.4	24.2 ± 0.1	28.6 ± 0.1	24.2 ± 0.3	23.2 ± 0.1	27.1 ± 0.2	33.2 ± 0.1	25.3 ± 0.2	33.2 ± 0.2	23.8 ± 0.2
O^3	46.5 ± 0.2	55.4 ± 0.1	62.1 ± 0.2	49.2 ± 0.2	57.2 ± 0.1	65.9 ± 0.1	68.2 ± 0.2	61.1 ± 0.2	45.4 ± 0.2	63.1 ± 0.2
PUMA	24.3 ± 0.2	30.5 ± 0.2	45.4 ± 0.2	27.9 ± 0.4	27.2 ± 0.2	31.1 ± 0.3	45.8 ± 0.2	38.0 ± 0.2	22.9 ± 0.3	63.5 ± 0.4
Single	27.8 ± 0.3	31.0 ± 0.2	49.8 ± 0.2	28.8 ± 0.1	26.0 ± 0.1	32.4 ± 0.2	46.3 ± 0.3	38.4 ± 0.2	24.6 ± 0.4	61.8 ± 0.2
Qwen3-8B										
Base	84.3 ± 0.2	75.7 ± 0.1	86.9 ± 0.2	79.1 ± 0.1	84.7 ± 0.3	78.1 ± 0.3	88.6 ± 0.2	78.0 ± 0.2	63.8 ± 0.3	74.5 ± 0.1
EUL	35.0 ± 0.2	32.2 ± 0.1	26.4 ± 0.2	41.6 ± 0.3	44.2 ± 0.2	41.2 ± 0.1	47.4 ± 0.2	45.1 ± 0.1	29.2 ± 0.2	54.4 ± 0.3
Arithmetic	37.4 ± 0.3	26.2 ± 0.2	21.0 ± 0.2	35.7 ± 0.1	43.2 ± 0.2	39.1 ± 0.1	28.9 ± 0.1	34.2 ± 0.2	29.9 ± 0.1	33.2 ± 0.2
TIES	37.8 ± 0.2	34.3 ± 0.2	27.3 ± 0.2	45.4 ± 0.1	42.3 ± 0.2	47.1 ± 0.1	35.2 ± 0.2	36.1 ± 0.1	33.7 ± 0.2	41.1 ± 0.1
MixLoRA	46.6 ± 0.2	36.7 ± 0.2	29.1 ± 0.2	51.7 ± 0.3	49.2 ± 0.3	39.2 ± 0.1	47.1 ± 0.3	31.7 ± 0.3	33.2 ± 0.1	44.1 ± 0.3
ALKN	25.5 ± 0.2	29.3 ± 0.1	24.2 ± 0.3	29.2 ± 0.1	25.8 ± 0.2	28.1 ± 0.1	34.2 ± 0.1	29.4 ± 0.2	32.5 ± 0.1	27.6 ± 0.2
O^3	59.1 ± 0.2	54.5 ± 0.2	64.2 ± 0.3	52.4 ± 0.1	52.6 ± 0.2	62.2 ± 0.2	64.1 ± 0.2	61.1 ± 0.1	42.3 ± 0.2	71.0 ± 0.1
PUMA	24.1 ± 0.2	42.3 ± 0.2	41.4 ± 0.2	23.6 ± 0.3	56.6 ± 0.1	32.1 ± 0.2	46.2 ± 0.2	42.9 ± 0.3	25.1 ± 0.2	71.4 ± 0.3
Single	25.3 ± 0.2	44.9 ± 0.3	43.3 ± 0.3	25.4 ± 0.1	57.3 ± 0.1	35.0 ± 0.2	49.3 ± 0.1	42.4 ± 0.3	25.8 ± 0.2	68.7 ± 0.2

accuracy on the evaluated forgetting domains while improving retained utility, showing that their benefits extend beyond stronger target suppression alone. Combining them gives the best forgetting and retention performance among the configurations examined. This supports their complementary roles: SE supplies information about the location of each parameter chunk, while MRH adapts the generated values to the projection module to which they belong.

Robustness to the number of training domains. We vary the number of domains contributing checkpoints to the shared training bank and evaluate the resulting generators on the same target domains. As shown in Table 3, forgetting and retention remain stable as the training bank covers more domains, with modest fluctuations rather than a consistent decline. Adding domains therefore does not substantially compromise the evaluated forgetting behaviors. Within the tested range, these results indicate that the generator is robust to the number of supervised domains and can accommodate broader checkpoint supervision while maintaining selective forgetting.

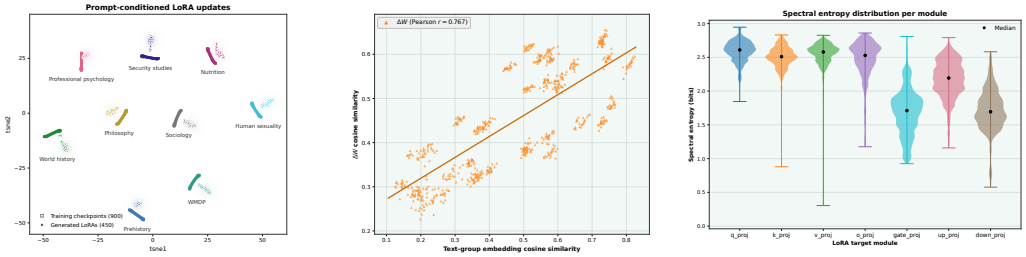
Qualitative analysis and discussion. Figure 2 examines how textual conditions and parameter structure relate to the generated updates. In panel (a), adapters generated from different domains occupy distinct regions of the projected parameter space, while panel (b) shows a positive association between text embedding similarity and effective-update similarity. Together, these patterns indicate that the generated parameters reflect the supplied textual conditions and support the interpretation

Table 2: Generator component ablation.

Variant	Forget domains ↓ Prehistory	Philosophy	Nutrition	Retain ↑ Average
Base	32.1	35.6	59.1	56.8
+ SE	31.0	34.3	58.3	57.7
+ MRH	30.8	33.4	57.4	58.2
+ SE + MRH	**29.3**	**31.8**	**55.6**	**59.4**

Table 3: Shared training domain ablation.

Domains	Forget domains ↓ Prehistory	Philosophy	Nutrition	Retain ↑ Average
3	28.8	31.9	56.3	58.6
5	29.5	31.6	55.1	59.2
7	29.2	30.4	55.5	59.6
9	29.3	31.8	55.6	59.4



(a) Prompt-conditioned updates (b) Text–update similarity (c) Module spectral entropy

Figure 2: Visual diagnostics of the parameter generator. (a) t-SNE visualization of prompt-conditioned LoRA updates, showing training checkpoints and generated adapters across nine domains. (b) Text-group embedding cosine similarity versus effective-update (ΔW) cosine similarity; the line shows the fitted trend. (c) Spectral entropy distributions across the seven LoRA target module types, with black dots marking medians.

of the generator as an implicit route from representative examples to domain-relevant unlearning behavior.

The generated adapters also occupy regions near, but not simply overlapping with, the corresponding training checkpoints in panel (a). This is consistent with learning a conditional mapping beyond reproducing individual reference weights. Considered alongside the improvements over Single, the visualization suggests that shared training can draw useful structure from the pooled domain trajectories to produce more effective adapters. The projection alone does not identify which domains contribute to a particular update, but it complements the behavioral evidence that parameter generation can improve on the independently optimized references.

Panel (c) reveals differences in spectral entropy across projection modules, indicating that their update spectra have different distributions. This motivates accounting for parameter structure within the generator: SE distinguishes chunks by their position and module identity, while MRH provides transformations specialized to each module type. Together with the component ablation, these observations support a generator that shares information across domains while accommodating differences within the adapter architecture.

5 Conclusion

We introduced PUMA to address the challenge of supporting unlearning requests across domains while preserving general model capabilities. PUMA learns a shared text-conditioned parameter generator from domain-specific LoRA checkpoints, allowing representative examples to guide the generation of unlearning adapters. This creates an implicit route from textual requests to LoRA weights and brings multi-domain supervision into a common framework. Once trained, the generator supports instance-based conditioning without further gradient optimization for each request. Across the evaluated backbones, PUMA improves both average forgetting efficacy and retained utility over independently trained domain-specific adapters. These findings demonstrate that learning to generate adapters can consolidate multi-domain unlearning while improving the balance between target suppression and general capability preservation. PUMA thus offers a promising direction for more flexible LLM unlearning, with generalization to unseen domains and finer-grained knowledge removal remaining important areas for future work.

6 AI USE STATEMENT

We only use AI to aid or polish writing.

REFERENCES

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pp. 141–159. IEEE, 2021.

Pengfei Cao, Chenhao Wang, Zhitao He, Hongbang Yuan, Jiachun Li, Yubo Chen, Kang Liu, Jun Zhao, et al. Rwku: Benchmarking real-world knowledge unlearning for large language models. *Advances in Neural Information Processing Systems*, 37:98213–98263, 2024.

Jiaao Chen and Diyi Yang. Unlearn what you want to forget: Efficient unlearning for llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12041–12052, 2023.

Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. *Advances in neural information processing systems*, 26, 2013.

Lior Deutsch. Generating neural networks with neural networks. *arXiv preprint arXiv:1801.01952*, 2018.

Vineeth Dorna, Anmol Mekala, Wenlong Zhao, Andrew McCallum, Zico Kolter, Zachary Lipton, and Pratyush Maini. Openunlearning: Accelerating llm unlearning via unified benchmarking of methods and metrics. *Advances in Neural Information Processing Systems*, 38, 2026.

Ronen Eldan and Mark Russinovich. Who's harry potter? approximate unlearning in llms. *arXiv preprint arXiv:2310.02238*, 2023.

Chongyu Fan, Jiancheng Liu, Licong Lin, Jinghan Jia, Ruiqi Zhang, Song Mei, and Sijia Liu. Simplicity prevails: Rethinking negative preference optimization for llm unlearning. *Advances in Neural Information Processing Systems*, 38:1540–1567, 2026.

Chongyang Gao, Lixu Wang, Kaize Ding, Chenkai Weng, Xiao Wang, and Qi Zhu. On large language model continual unlearning. In *International Conference on Learning Representations*, volume 2025, pp. 101772–101801, 2025.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 32, 2019.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

David Ha, Andrew Dai, and Quoc V. Le. HyperNetworks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=rkpACe1lx`.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. Knowledge unlearning for mitigating privacy risks in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14389–14408, 2023.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, et al. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. *arXiv preprint arXiv:2404.15159*, 2024a.

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, et al. The wmdp benchmark: Measuring and reducing malicious use with unlearning. *arXiv preprint arXiv:2403.03218*, 2024b.

Zhiyuan Liang, Dongwen Tang, Yuhao Zhou, Xuanlei Zhao, Mingjia Shi, Wangbo Zhao, Zekai Li, Peihao Wang, Konstantin Schürholt, Damian Borth, et al. Drag-and-drop llms: Zero-shot prompt-to-weights. *Advances in Neural Information Processing Systems*, 38:153258–153283, 2026.

Yujian Liu, Yang Zhang, Tommi Jaakkola, and Shiyu Chang. Revisiting who's harry potter: Towards targeted unlearning from a causal intervention perspective. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8708–8731, 2024.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 565–576, 2021.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C Lipton, and J Zico Kolter. Tofu: A task of fictitious unlearning for llms. *arXiv preprint arXiv:2401.06121*, 2024.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

Martin Pawelczyk, Seth Neel, and Himabindu Lakkaraju. In-context unlearning: Language models as few shot unlearners. *arXiv preprint arXiv:2310.07579*, 2023.

William Peebles, Ilija Radosavovic, Tim Brooks, Alexei A Efros, and Jitendra Malik. Learning to learn with generative models of neural network checkpoints. *arXiv preprint arXiv:2209.12892*, 2022.

Jason Phang, Yi Mao, Pengcheng He, and Weizhu Chen. Hypertuning: Toward adapting large language models without back-propagation. In *International Conference on Machine Learning*, pp. 27854–27875. pmlr, 2023.

Jose Miguel Lara Rangel, Stefan Schoepf, Jack Foster, David Krueger, and Usman Anwar. Learning to forget using hypernetworks. *arXiv preprint arXiv:2412.00761*, 2024.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021.

Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah Smith, and Chiyuan Zhang. Muse: Machine unlearning six-way evaluation for language models. In *International Conference on Learning Representations*, volume 2025, pp. 27797–27818, 2025.

Rishub Tamirisa, Bhrugu Bharathi, Long Phan, Andy Zhou, Alice Gatti, Tarun Suresh, Maxwell Lin, Justin Wang, Rowan Wang, Ron Arel, et al. Tamper-resistant safeguards for open-weight llms. In *International Conference on Learning Representations*, volume 2025, pp. 101802–101829, 2025.

Kai Wang, Dongwen Tang, Boya Zeng, Yida Yin, Zhaopan Xu, Yukun Zhou, Zelin Zang, Trevor Darrell, Zhuang Liu, and Yang You. Neural network diffusion. *arXiv preprint arXiv:2402.13144*, 2024.

Kai Wang, Dongwen Tang, Wangbo Zhao, Konstantin Schürholt, Zhangyang Wang, and Yang You. Recurrent diffusion for large-scale parameter generation. *arXiv preprint arXiv:2501.11587*, 2025.

Piotr Wójcik, Maksym Petrenko, Wojciech Gromski, Przemysław Spurek, and Maciej Zieba. Unhype: Clip-guided hypernetworks for dynamic lora unlearning. In *Forty-third International Conference on Machine Learning*, 2026.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. Pmlr, 2022.

Abudukelimu Wuerkaixi, Qizhou Wang, Sen Cui, Wutong Xu, Bo Han, Gang Niu, Masashi Sugiyama, and Changshui Zhang. Adaptive localization of knowledge negation for continual llm unlearning. In *Forty-second International Conference on Machine Learning*, 2025.

Xiaoyu Xu, Minxin Du, Kun Fang, Yaxin Xiao, Zhicong Huang, Cheng Hong, Qingqing Ye, and Haibo Hu. Fit to forget: Robust continual unlearning for large language models. *arXiv preprint arXiv:2601.21682*, 2026a.

Xiaoyu Xu, Minxin Du, Zitong Li, Zi Liang, Zhibiao Guo, Shiyu Zhang, Peizhao Hu, Qingqing Ye, and Haibo Hu. From domains to instances: Dual-granularity data synthesis for LLM unlearning. *arXiv preprint arXiv:2601.04278*, 2026b. URL https://arxiv.org/abs/2601.04278.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115, 2023.

Yuanshun Yao and Xiaojun Xu. Large language model unlearning. *Advances in Neural Information Processing Systems*, 37:105425–105475, 2024.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. *arXiv preprint arXiv:2404.05868*, 2024.

A Experimental Details

We construct a separate source-adapter bank and train a dedicated parameter generator for each evaluated backbone. Source adapters are trained with the forgetting and retention objectives in Equation 7, while the backbone remains frozen. The generator is subsequently trained to reconstruct adapter parameters from domain-specific text conditions, without backpropagation through the backbone.

Adapter supervision and optimization. The supervision bank contains the first 100 continuation checkpoints from each domain-specific adapter trajectory. Each training pair combines a checkpoint with 64 texts sampled without replacement from the corresponding domain's conditioning pool. The text encoder remains frozen and in evaluation mode during both training and generation. Table 4 summarizes the training configuration. We optimize the generator with 8-bit AdamW and clip the gradient norm during training. The decoder and structural components are updated jointly, while the source checkpoints provide fixed reconstruction targets. Backbone-dependent output layouts are described in Appendix B.

Evaluation and aggregation. For Single and PUMA, each requested domain has its own adapter. Its corresponding target accuracy supplies the domain's entry in Table 1, and Retain is averaged over both the 49 retained MMLU tasks and the nine requested-domain adapters. The resulting score summarizes performance across domain-specific requests. The Biology and Cybersecurity score is an equal-weight average of the two WMDP task accuracies. Across-task averages give equal weight to tasks rather than to individual questions. Condition-group variation measures sensitivity to the supplied texts, whereas independent generator runs measure training variability. Checkpoints from the same trajectory share a training run.

Deployment configurations and evaluation splits. The archived composition experiments use checkpoint-100 source adapters. Arithmetic sums their effective updates with a multiplier of 0.25, while TIES applies top-20% trimming, sign resolution, and disjoint averaging with a multiplier of 1.5. The archived joint LoRA configuration uses checkpoint 100 and a multiplier of 1.125. These values specify the recorded baseline configurations.

The archived generator experiments include a raw-target DnD decoder evaluated at step 3000 and controlled generator comparisons at step 5000. The former selects per-domain update multipliers using full-benchmark scores; the latter separates approximately 20% of each task for calibration and 80% for testing through a document-hash partition. For calibrated evaluations, the selected multiplier acts on the effective update as $W'_\ell = W_{0,\ell} + s\Delta W_\ell$. For $s > 0$, scaling both LoRA factors by $\sqrt{s}$ produces the same update. This selection uses labeled evaluation data without gradient optimization of the generated adapter. The experiment records identify the backbone, generator and adapter checkpoints, conditioning groups, deployment multiplier, evaluator, and selection split for each configuration.

B Generator Architecture

PUMA combines a frozen all-MiniLM-L12-v2 text encoder with a hyper-convolutional parameter decoder, following DnD (Liang et al., 2026). Each conditioning group contains 64 texts, represented by token features of shape $64 \times 1024 \times 384$. The encoder processes each text in segments of up to 512 tokens and concatenates their hidden states to form the decoder input. Attention is computed within each segment.

Decoder and parameter representation. The decoder transforms the text features through the following sequence of tensor layouts:

$$\begin{aligned}(64, 1024, 384) &\to (64, 500, 300) \to (256, 125, 300) \\ &\to (1024, 64, 300) \to (2048, 16, 256) \to (N, h, w),\end{aligned} \tag{13}$$

with convolution kernel size 11. The output layout follows the parameter tokenizer: N is the number of adapter chunks, and $h \times w$ is the shape of each chunk, including boundary metadata and padding. The tokenizer determines this layout from the LoRA factors of the target backbone. Each chunk

Table 4: Source-adapter and parameter-generator training settings. Optimization settings refer to generator training unless otherwise specified.

Setting	Value
Source LoRA rank / alpha	8 / 8
Source LoRA target modules	`all-linear`
Source LoRA dropout	0.05
Source checkpoints per domain	100
Texts per conditioning group	64
Maximum tokens per condition text	1,024
Generator optimizer	8-bit AdamW
Learning rate / weight decay	10^{-5} / 0.1
Global batch size	8
Warmup / total optimizer steps	100 / 5,000
Checkpoint saving interval	1,000 steps
Gradient norm clipping	1.0
Generator initialization / condition-split seed	999 / 999

stores standardized factor values together with its mean and standard deviation, allowing the decoder output to be converted back into adapter weights. Normalization and metadata decoding use high precision. Padding is excluded from the reconstruction loss, whose channel weights reflect variation across source checkpoints.

Structural embeddings and module-specific residual heads. The structural embedding (SE) branch represents each chunk through its layer, module type, factor identity, and within-factor position. These indices select embeddings of dimension $d_s = 8$, which are summed and mapped to a chunk-shaped correction by a shared bias-free projection. Let L, U, F, and C denote the numbers of layer, module, factor, and chunk-position entries, respectively. The adapter layout determines these cardinalities, and the SE parameter count is

$$P_{\mathrm{SE}} = d_s(L + U + F + C) + d_s hw. \tag{14}$$

The module-specific residual heads (MRH) refine these corrected chunks according to their projection type. A separate head is assigned to each of the query, key, value, output, gate, up, and down projections. Each head applies non-affine layer normalization, a linear map from w to a bottleneck of width $d_h = 16$, SiLU, and a linear map back to w. Both linear maps include biases. The transformation is applied row-wise, with weights shared across layers and LoRA factors of the same module type. The resulting parameter count is

$$P_{\mathrm{MRH}} = U(2wd_h + d_h + w). \tag{15}$$

The total number of added parameters is $P_{\mathrm{SE}} + P_{\mathrm{MRH}}$. These expressions account for the backbone-dependent adapter layout without assigning the same parameter count to every generator.

The SE projection and the output projection and bias of each residual head are initialized to zero. The augmented generator therefore starts with the underlying decoder's predictions and learns structural and module-specific corrections through joint training.

Functional reconstruction. LoRA factors admit equivalent parameterizations: for any invertible rank-space transformation Q, the factors $A' = QA$ and $B' = BQ^{-1}$ satisfy $B'A' = BA$. Factor-space reconstruction measures the agreement with a particular checkpoint representation, whereas effective-update similarity measures the agreement between the induced weight changes. Behavioral evaluation complements these parameter-space diagnostics by measuring target suppression and retained utility after the generated adapter is applied.

C DATASETS

We consider nine forgetting domains. Eight correspond to MMLU subjects (Hendrycks et al., 2020), and the ninth combines the biology and cybersecurity subsets of WMDP (Li et al., 2024b). Table 5 lists their evaluation tasks. WMDP chemistry is excluded from the nine-domain forgetting aggregate and from Retain49, even when it is available in the underlying evaluation records.

Table 5: Forgetting domains and their evaluation tasks. The combined WMDP domain contributes one equally weighted domain score.

Forgetting domain	Evaluation task
Prehistory	MMLU: Prehistory
Professional Psychology	MMLU: Professional Psychology
High School World History	MMLU: High School World History
Philosophy	MMLU: Philosophy
Human Sexuality	MMLU: Human Sexuality
Nutrition	MMLU: Nutrition
Sociology	MMLU: Sociology
Security Studies	MMLU: Security Studies
Biology and Cybersecurity	WMDP: Biology and Cybersecurity

Conditioning data and benchmark separation. The MMLU-domain conditioning corpora contain constructed domain texts rather than the official MMLU evaluation questions. The recorded condition split uses seed 999 and reserves 100 texts per domain for generation. For a pool of 500 texts, this leaves 400 training conditions. Each generation group contains 64 texts, and groups may overlap. This is a condition holdout within supervised domains: all nine domains contribute checkpoints to the full generator's training bank.

Record-level separation does not by itself establish textual disjointness. The archived WMDP condition split contains three exact duplicates across its training and holdout pools. Moreover, holding a text out of generator conditioning does not establish that it was absent from source-adapter training. We therefore distinguish condition holdout from unseen-domain generalization and from removal of specific training records.

Retained knowledge. Retain49 comprises the 49 MMLU subjects outside the eight designated MMLU forgetting subjects. This fixed set permits consistent aggregation across requests. However, it does not measure collateral effects on the other candidate forgetting domains when only one domain is requested. The cross-domain matrices in Appendix E complement Retain49 by exposing these effects directly.

D Baseline Methods

The baselines cover independent domain adaptation, joint optimization, adapter composition, expert routing, and continual unlearning. All results use the forgetting-domain and Retain49 metrics defined in the main text. The descriptions below distinguish the original methods from their implementations in our multi-domain setting.

Base. Base is the released backbone without an unlearning update. Its scores establish the initial target knowledge and general capability against which forgetting and retention are evaluated.

Single. Single trains an independent LoRA adapter for each forgetting domain using the EUL-inspired objective in Equation 7. Each request loads the corresponding domain adapter. Target scores come from the respective adapters, and Retain averages their retained utility. This baseline provides the domain-specific reference for evaluating whether shared parameter generation preserves or improves independently optimized forgetting behavior.

EUL. EUL introduces lightweight unlearning layers trained with a selective teacher–student objective and combines them through learned fusion (Chen & Yang, 2023). Our EUL baseline adapts its forgetting and retention objectives to causal language modeling with LoRA. The row labeled EUL uses a jointly trained adapter over the pooled forgetting domains, corresponding to the joint LoRA configuration in the experiment records. Single instead applies the objective independently to each domain. This comparison evaluates direct joint optimization against domain-specific training and shared parameter generation.

Arithmetic. Task Arithmetic composes task-specific parameter changes by adding their task vectors (Ilharco et al., 2022). We apply this operation to the effective updates of the domain-specific unlearning adapters. For module ℓ, the combined update is $s \sum_{d \in \mathcal{D}} \Delta W_{d,\ell}$, where s is the deployment multiplier. Composition operates on the effective LoRA products rather than on the factors separately, producing a fixed update shared across requests.

TIES. TIES-Merging addresses interference through trimming, sign selection, and merging (Yadav et al., 2023). We construct task vectors from the domain-specific effective updates, remove small-magnitude entries, resolve sign disagreements, and average the surviving entries aligned with the selected sign. The resulting merged update is applied to the common backbone. This baseline tests whether resolving conflicts directly in parameter space preserves multi-domain forgetting and retained utility.

MixLoRA. MixLoRA combines LoRA experts in feed-forward blocks with an input-dependent router (Li et al., 2024a). We adapt this architecture to joint unlearning by training its experts and router on pooled domain data with the forgetting and retention objectives. Attention projections use conventional LoRA adapters, while feed-forward projections use routed low-rank experts. The reference implementation matches the total LoRA-factor capacity to the joint LoRA baseline and accounts for routing parameters separately. Unlike PUMA, MixLoRA routes computation among trained experts during model execution.

ALKN. Adaptive Localization of Knowledge Negation (ALKN) performs continual unlearning through task-vector negation, dynamically masked gradients, and adaptive update intensity (Wuerkaixi et al., 2025). Our implementation processes the domain requests sequentially and applies the localized updates to the backbone parameters. We evaluate the resulting model after the complete request sequence. ALKN therefore represents continual parameter updating, with its trainable parameterization extending beyond the LoRA factors used by the adapter baselines.

O^3. O^3 combines orthogonal unlearning LoRA adapters with an out-of-distribution detector (Gao et al., 2025). Orthogonality separates updates associated with successive requests, while the detector controls whether and to what extent an adapter is activated for an input. We apply this framework to the sequence of domain-level forgetting requests and evaluate the final system with its detector-based routing. This provides a comparison between explicit routing over learned unlearning adapters and PUMA's implicit text-to-parameter mapping.

E More Experimental Results

E.1 Results on Additional Models

Table 6 extends the evaluation to the Llama backbone using the same forgetting and retention protocol as the main comparison. PUMA again achieves stronger average forgetting and higher retained utility than Single, showing that the shared generator can improve on independently trained domain-specific adapters in another model family. It supports the same one-for-all, instance-conditioned interface: representative texts determine the generated adapter without additional gradient optimization for each request.

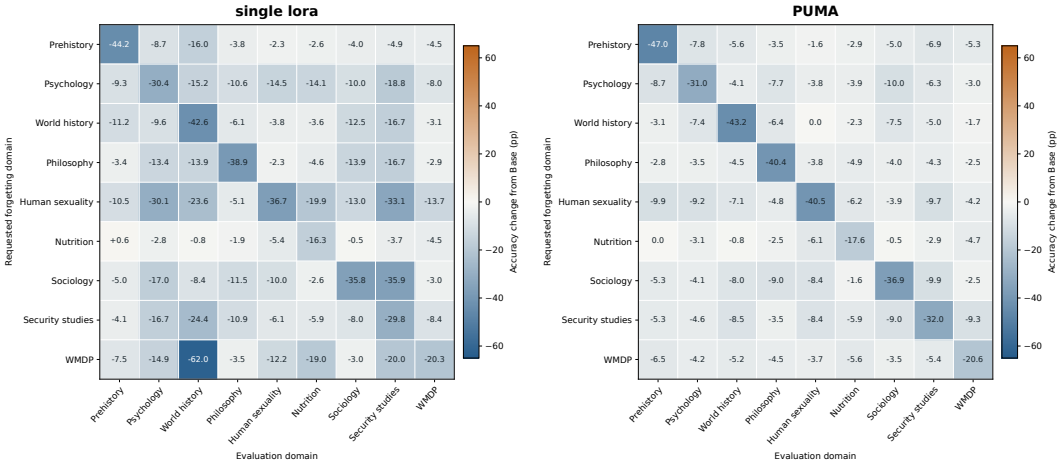
The baseline comparison follows the pattern observed in the main results. ALKN suppresses target accuracy at the cost of severe retention degradation, while Arithmetic and TIES also incur substantial losses in general capability. O^3 preserves more utility but leaves considerably higher target accuracy than PUMA. PUMA combines effective target suppression with the highest retained utility among the compared methods on this backbone. Together with the Qwen results, these findings demonstrate the effectiveness of the proposed generation framework across the evaluated model families and sizes.

E.2 Cross-Domain Forgetting and Retention

Figure 3 compares the cross-domain behavior of PUMA with Single. Rows identify the requested forgetting domain, columns identify the evaluated domain, and each cell shows the accuracy change

Table 6: Forgetting and retained utility on Llama-3.1-8B-Instruct (accuracy, %). Retain is the mean over the same 49 MMLU tasks used in Table 1. Biology and Cybersecurity averages the two WMDP tasks.

	Forget domains ↓									Retain ↑
Method	Prehistory	Professional Psychology	High School World History	Philosophy	Human Sexuality	Nutrition	Sociology	Security Studies	Biology and Cybersecurity	Average
					Llama-3.1-8B-Instruct					
Base	75.9 ± 0.2	72.1 ± 0.1	86.1 ± 0.3	72.7 ± 0.2	79.4 ± 0.1	76.1 ± 0.2	83.6 ± 0.2	75.5 ± 0.3	59.7 ± 0.2	66.8 ± 0.3
EUL	43.6 ± 0.2	31.8 ± 0.3	49.2 ± 0.3	54.9 ± 0.2	55.7 ± 0.3	35.2 ± 0.2	67.1 ± 0.2	62.3 ± 0.1	28.2 ± 0.2	48.1 ± 0.1
Arithmetic	43.8 ± 0.2	25.2 ± 0.1	47.1 ± 0.2	44.9 ± 0.2	54.2 ± 0.2	31.9 ± 0.1	47.4 ± 0.2	46.2 ± 0.1	25.8 ± 0.2	32.5 ± 0.1
TIES	44.4 ± 0.1	27.2 ± 0.1	51.0 ± 0.2	55.9 ± 0.2	48.4 ± 0.2	42.7 ± 0.1	55.4 ± 0.2	53.9 ± 0.1	29.2 ± 0.3	39.1 ± 0.2
MixLoRA	53.6 ± 0.1	29.7 ± 0.2	51.0 ± 0.1	63.1 ± 0.2	58.8 ± 0.1	34.2 ± 0.2	65.2 ± 0.3	48.2 ± 0.1	28.7 ± 0.3	43.0 ± 0.1
ALKN	33.1 ± 0.1	18.2 ± 0.2	51.4 ± 0.2	35.2 ± 0.2	36.2 ± 0.1	24.2 ± 0.1	53.1 ± 0.1	44.5 ± 0.2	20.2 ± 0.2	21.1 ± 0.1
O^3	59.0 ± 0.4	43.9 ± 0.2	65.3 ± 0.2	50.4 ± 0.2	55.1 ± 0.2	51.6 ± 0.1	65.4 ± 0.2	57.6 ± 0.2	35.6 ± 0.2	60.1 ± 0.1
PUMA	33.6 ± 0.3	31.5 ± 0.2	39.6 ± 0.2	37.5 ± 0.1	51.2 ± 0.3	34.4 ± 0.2	45.6 ± 0.2	48.7 ± 0.3	23.1 ± 0.1	63.1 ± 0.2
Single	36.1 ± 0.2	32.8 ± 0.3	40.9 ± 0.1	39.9 ± 0.2	51.9 ± 0.1	36.3 ± 0.3	48.2 ± 0.2	49.6 ± 0.2	24.6 ± 0.3	61.4 ± 0.2



(a) Single: domain-specific LoRA

(b) PUMA: generated LoRA

Figure 3: Cross-domain accuracy changes relative to Base on Qwen2.5-3B (percentage points). Rows identify the requested forgetting domain and columns identify the evaluated domain. Both panels use the same color scale. Blue denotes reduced accuracy and orange denotes increased accuracy. The diagonal measures target suppression, while off-diagonal cells reveal changes in non-requested domains. WMDP denotes the combined biology and cybersecurity domain.

relative to Base. Diagonal decreases indicate suppression of the requested knowledge, while off-diagonal changes reveal effects on non-requested domains. Each row corresponds to a separately applied adapter.

Compared with Single, PUMA achieves stronger target suppression while reducing substantial collateral losses across several non-requested domains. The heatmaps show a more selective pattern: accuracy decreases remain concentrated on the requested targets, and many off-diagonal entries move closer to the unchanged model's performance. This is particularly evident for WMDP and Human Sexuality requests, where Single affects knowledge well beyond the intended target and PUMA better preserves the surrounding domains.

These results show that PUMA's advantage extends beyond the aggregate retained-utility score. It better separates the knowledge designated for forgetting from other knowledge represented in the same multi-domain training bank. Shared parameter generation therefore improves the overall balance between forgetting and preservation compared with independently trained domain adapters, supporting its use for selective multi-domain unlearning.